Job shop scheduling: Operations

- Task is to schedule a set of *jobs* subject to a set of *constraints*
- Each job has a *process plan*
  - Consisting of *operations* needed to complete the job
  - Each operation, i, has a *processing time*, p_i
  - *Sequencing constraints* between operations
    - If operation i precedes operation j, denoted $i \rightarrow j$, then

$$st_i + p_i \leq st_j$$

- Each job J has a *ready time*, r_J, and a *deadline*, d_J
  - For each operation i of job J

$$r_J \leq st_i$$

$$st_i + p_i \leq d_J$$

Job shop scheduling: Resources

- Operations need *resources*
  - At most one operation may use a resource at any given time
- If operations i and j require the same resource, then

$$st_i + p_i \leq st_j \quad or \quad st_j + p_j \leq st_i$$

Search framework

- View problem as that of establishing *sequencing constraints* between pairs of operations that share a common resource

- For operations i and j that share a common resource
  - Decision $i \rightarrow j$ leads to constraint
    $$st_i + p_i \leq st_j$$
  - Decision $j \rightarrow i$ leads to constraint
    $$st_j + p_j \leq st_i$$

- At the start and after each decision, compute the *earliest* (est_i) and *latest* (lst_i) *start time* of each operation i

Ordering of operations

- Case 1: If $est_i + p_i \leq lst_j$ and $est_j + p_j > lst_i$
- Case 2: If $est_i + p_i > lst_j$ and $est_j + p_j \leq lst_i$
- Case 3: If $est_i + p_i > lst_j$ and $est_j + p_j > lst_i$
- Case 4: If $est_i + p_i \leq lst_j$ and $est_j + p_j \leq lst_i$

Search procedure

- Initialize start time bounds using *bellman-ford* on the distance graph *G* resulting from the operations constraints
- Select an unsequenced pair of operations that require the same resource, select a sequence, and propagate the constraint
  – First select operations that satisfy Cases 1 or 2
  – Select among pairs that satisfy Case 4 using the variable and value ordering heuristic
  – Backtrack if an inconsistency is detected
    - Resulting distance graph contains negative cycles
    - Operations that satisfy Case 3 are detected

Variable and value ordering heuristics

- For unordered operations i, j that share a resource, define the *temporal slack:*
  - $Slack(i \to j) = lst_j - (est_i + p_i)$
  - $Slack(j \to i) = lst_i - (est_j + p_j)$
- *Overall slack* of a decision is
  - $Min(Slack(i \to j), Slack(j \to i))$
- Variable ordering heuristic: Pick the decision with the *minimum* overall slack
- Value ordering heuristic:
  - Select $i \to j$ if $Slack(i \to j) \geq Slack(j \to i)$
  - Select $j \to i$ otherwise

Constraint propagation

- Adding decision $i \rightarrow j$ to the distance graph G corresponds to adding and edge from j to i with weight $-p_i$
- d_{0i} and d_{i0} can be computed
  - *Directly* using *bellman-ford*
  - *Incrementally* using constraint propagation
    - Relaxing an edge from j to i can update distance to i
    - If the distance to i is updated, then one needs to relax every edge (i, k)
    - Stop propagation if
      - For some node i, $est_i > lst_i$ or
      - The original edge is successfully relaxed twice